Um mecanismo orientado a ISP para escolha tendenciosa de pares no BitTorrent

Alejandra Klachquin[1], **Daniel R. Figueiredo**[1]

[1]Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ
Caixa Postal 68.511 – 21.941-972 – Rio de Janeiro, RJ – Brasil

alejandra@land.ufrj.br, daniel@land.ufrj.br

Abstract. *BitTorrent is one of the most popular peer-to-peer (P2P) file sharing applications used on the Internet and is responsible for a significant amount of traffic between ISPs (Internet Service Provider) networks. A big challenge for ISPs today is controlling traffic generated by P2P applications, whose traffic pattern depends directly on the P2P network. Neighbor selection mechanisms play a central role in P2P network structure as it determines with which peers a given peer will communicate. In this work, we propose a simple biased neighbor selection mechanism that considers the location of peers with respect to a single ISP. A theoretical analysis and detailed BitTorrent simulations show the advantages and disadvantages of this mechanism. To mitigate its negative effects, we propose a simple modification to the BitTorrent protocol, which impose the peers to stay connected a small amount of time after completing the download.*

Resumo. *O BitTorrent é um dos aplicativos peer-to-peer (P2P) mais populares da Internet e responsável por boa fração de tráfego nas redes dos ISPs (Internet Service Providers). Um dos grandes desafios dos ISPs está em controlar o tráfego gerado por aplicativos P2P, cujo padrão de tráfego é influenciado diretamente pela estrutura da rede P2P. O mecanismo de seleção de vizinhos possui papel central na formação da topologia da rede, pois determina com quais outros peers um determinado peer irá se comunicar. Neste trabalho apresentamos uma proposta simples para seleção tendenciosa de vizinhos, que leva em consideração a localização dos peers com relação a um único ISP. Através de uma avaliação teórica e de resultados de simulação detalhada do BitTorrent, mostramos as vantagens e desvantagens deste mecanismo. Para mitigar seus efeitos negativos, apresentamos uma simples modificação no protocolo BitTorrent, onde os peers passam a ser obrigados a permanecer um pequeno período de tempo a mais conectados após concluírem o download.*

1. Introdução

Aplicativos baseados na arquitetura *peer-to-peer* (P2P) vêm tendo um grande impacto na Internet, não somente pela variedade de aplicativos e enorme número de usuários, mas também pela grande quantidade de tráfego gerada por alguns destes aplicativos. Por exemplo, o BitTorrent (BT) é um aplicativo para compartilhamento de arquivos usado diariamente por milhões de usuários e responsável por uma boa fração do tráfego nas redes dos ISPs (Internet Service Providers) [H. Schulze and K. Mochalski 2009].

Neste contexto (e.g., BT), usuários correspondem a *peers* que formam uma rede e participam ativamente do processo de disseminação de conteúdo; baixando e subindo

blocos de um arquivo entre si. Como todos estes usuários pertencem a algum ISP (*Internet Service Provider*), o tráfego gerado pelo aplicativo necessariamente atravessa as redes dos ISPs, dando origem a um grande volume de dados.

Um dos maiores desafios enfrentados por ISPs de hoje está na redução do tráfego P2P em suas redes, principalmente nos enlaces entre ISPs, que são geralmente mais custosos e possuem uma maior utilização. Diversas técnicas vêm sendo propostas para atacar este problema e podemos dividi-las em três categorias: *caching* de conteúdo, *shaping* de tráfego, e modificação na aplicação [Sandvine Co. 2004, Saleh and Hefeeda 2006, Shen et al. 2007, Aggarwal et al. 2007, Bindal et al. 2006, Choffnes and Bustamante 2008, Xie et al. 2008].

A ideia de *caching* é armazenar conteúdo pertinente à aplicação P2P dentro do ISP e redirecionar os pedidos dos *peers* internos ao ISP a este repositório. Neste caso, o ISP deve manter e operar esta infraestrutura, que além do custo e das dificuldades técnicas, pode acarretar em problemas legais (por exemplo, se o conteúdo armazenado for ilegal). A ideia de *shaping* de tráfego é simplesmente policiar e limitar a banda para o tráfego P2P nos enlaces entre IPSs. Por fim, a última técnica se baseia na modificação dos aplicativos P2P para que os mesmos tenham conhecimento dos ISPs e tomem decisões neste sentido.

Parte do problema do tráfego gerado pelos aplicativos P2P é que eles não consideram o contexto ou as preferências dos ISPs em seu processo de seleção de *peers* para solicitar/oferecer serviço. Por exemplo, um *peer* do BT pode igualmente solicitar um bloco de dados tanto a um *peer* interno ao seu ISP quanto a um *peer* externo. Entretanto, seria melhor para o ISP se UM *peer* interno fosse escolhido, reduzindo assim a quantidade de tráfego pelo enlace entre ISPs.

Existem algumas propostas na literatura para modificação dos aplicativos de forma a levarem em consideração a localização dos *peers* com relação aos ISPs [Aggarwal et al. 2007, Bindal et al. 2006]. No caso do BT, existem propostas de modificação apenas de uma entidade central, chamada de *tracker* (descrita na Seção 2) [Bindal et al. 2006]. Esta última abordagem é bastante vantajosa pois funciona independente das muitas variações dos aplicativos BT utilizadas pelos usuários. Entretanto, tal modificação pode trazer vantagens e desvantagens para os usuários do BT assim como para o ISP.

Neste artigo, iremos propor uma técnica simples que modifica o comportamento do *tracker* no BT com o objetivo de reduzir o tráfego entre ISPs. Nossa técnica é inspirada na proposta de Bindal et. al [Bindal et al. 2006], porém mais fácil de ser implementada, não sendo necessário a cooperação entre ISPs (mais detalhes na Seção 3). Avaliamos a técnica proposta caracterizando o desempenho tanto dos usuários (i.e., tempo médio de *download*), quanto do ISP (i.e., redução no tráfego inter-ISP). A avaliação é feita primeiro de forma aproximada, utilizando um modelo analítico simplificado, e em seguida utilizando um simulador detalhado do BT. Nossos resultados indicam que a escolha dos parâmetros da técnica possui um impacto fundamental em seu desempenho, podendo ser extremamente prejudicial para os usuários em alguns casos. Em seguida, propomos uma simples modificação no BT que oferece um desempenho superior aos usuários sem denegrir as vantagens obtidas pelo ISP.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 descrevemos o funcionamento do BT e na Seção 3 apresentamos nossa proposta para escolha tendenciosa de *peers*. Uma avaliação simplificada do processo de seleção de vizinhos é apresentada na Seção 4. O simulador utilizado na avaliação da proposta assim como as medidas de interesse estão descritas na Seção 5. Os resultados obtidos são apresentados na Seção 6. A proposta de modificação no BT é apresentada na Seção 7 juntamente com seus resultados. Finalmente, na Seção 8, apresentamos a conclusão do trabalho.

2. BitTorrent e trabalhos relacionados

Para descrever o funcionamento do BT, iremos primeiramente definir os componentes que constituem o aplicativo e a rede que o mesmo forma para distribuir conteúdo:

1. *Peer*: um usuário do aplicativo BT (na verdade, seu computador).
2. *Swarm*: é o conjunto de *peers* que estão participando da distribuição de um determinado conteúdo.
3. *Seed*: é um *peer* que possui um determinado conteúdo por completo.
4. *Leecher*: é um *peer* que possui apenas parte de um conteúdo.
5. *Tracker*: é um servidor que mantém uma lista com todos os *peers* presentes no *swarm*. Entre outras atribuições, o *tracker* deve enviar uma lista de *peers* que já fazem parte do *swarm* a todo *peer* que se junta ao *swarm*. Além disso, os *peers* periodicamente enviam informações a respeito do andamento do *download* ao *tracker* e eventualmente podem requisitar mais *peers*.
6. *Torrent*: é um pequeno arquivo que contém metadados a respeito do conteúdo a ser distribuído, além de possuir o endereço (IP ou URL) de um ou mais *trackers* que irão coordenar seus respectivos *swarms*.
7. *Pedaços*: o conteúdo a ser distribuído é dividido em pedaços de tamanho fixo (e.g., 128KB), que por sua vez são divididos em blocos (e.g., 16KB). Ao adquirir todos os blocos referentes a um determinado pedaço, um *peer* torna-se apto a disseminá-los.

Tendo em vista essa terminologia, a rede de distribuição é criada a partir de um usuário, o *seed* original, que irá publicar o conteúdo na rede através do *torrent*. *Peers* interessados no conteúdo obtém o arquivo *torrent* e, com isso, contactam o *tracker*, o qual irá adicioná-los ao *swarm* e enviar-lhes uma lista contendo o endereço de no máximo L *peers* participantes do mesmo *swarm*. Para selecionar esses *peers*, o *tracker* utiliza uma política completamente aleatória. Em particular, a probabilidade de um determinado *peer* do *swarm* ser escolhido é igual para todo *peer*, não sendo levado em consideração nenhum fator diferenciador entre os *peers*. Um novo *peer* estabelece conexão com os *peers* da lista recebida do *tracker*, tornando-se vizinhos. É apenas através destes vizinhos que um *peer* irá obter e oferecer blocos do conteúdo sendo distribuído. Por fim, caso o número de vizinhos de um *peer* encontre-se abaixo de um determinado valor, uma nova lista de *peers* é requisitada ao *tracker*. Mais detalhes sobre o funcionamento do BT podem ser encontrados na literatura.

Repare que a seleção dos vizinhos feita pelo *tracker* original do BT é totalmente aleatória e não utiliza qualquer critério de localização do *peer*. Um dos motivos para esta política é dar maior diversidade na rede P2P formada pelos vizinhos, deixando a rede mais conectada. Em contrapartida, esta mesma política pode gerar grandes distorções

com relação aos vizinhos de um *peer* e seus respectivos ISPs. Por exemplo, considere um *swarm* formado por *peers* pertencentes a apenas dois ISPs, cada um com o mesmo número de *peers*. Intuitivamente, metade dos vizinhos de um *peer* estará localizada no ISP diferente ao seu. Dessa forma, muitos blocos serão trocados entre os *peers* de ISPs distintos, o que é ruim para os ISPs, pois aumenta o tráfego nos enlaces inter-ISPs que são geralmente mais custosos e mais sobrecarregados.

Com o objetivo de evitar estes cenários, diversos mecanismos de seleção de vizinhos foram propostos na literatura [Bindal et al. 2006, Aggarwal et al. 2007, Choffnes and Bustamante 2008, Lin et al. 2010, Le-Blond et al. 2010]. Esses métodos levam em consideração diversas informações pertinentes aos *peers* e aos ISPs de forma a realizar uma escolha de vizinhos que seja mais apropriada, ou seja, que atenda aos interesses do ISP e dos usuários. Desta forma, a política de seleção deixa de ser aleatória e passa a ser o que chamamos de tendenciosa.

Em [Lin et al. 2010], o protocolo BT é modificado de forma que um *peer* somente requisita um pedaço a *peers* externos ao seu ISP caso seus vizinhos internos não possuam esse pedaço. Por outro lado, [Bindal et al. 2006] e [Le-Blond et al. 2010] modificam o *tracker* de forma a limitar o número de *peers* externos que os usuários do ISP podem conectar-se. [Aggarwal et al. 2007] e [Choffnes and Bustamante 2008] apresentam uma proposta similar, porém os *peers* são classificados através de um oráculo, que prevê, por exemplo, a distância entre os *peers*. Um estudo complementar a estes mecanismos é realizado em [Wang et al. 2010], que apresenta um método para estabelecer quantos e quais *trackers* devem ser modificados com maior prioridade, a fim de implementar a seleção tendenciosa de pares com maior eficiência. Neste trabalho, apresentamos uma proposta similar a [Bindal et al. 2006] e [Le-Blond et al. 2010], porém mais simples. Em nosso mecanismo não é necessária a utilização de informações que escapam ao domínio do ISP e tampouco requer conhecer o número de conexões entre *peers* internos e externos ao ISP.

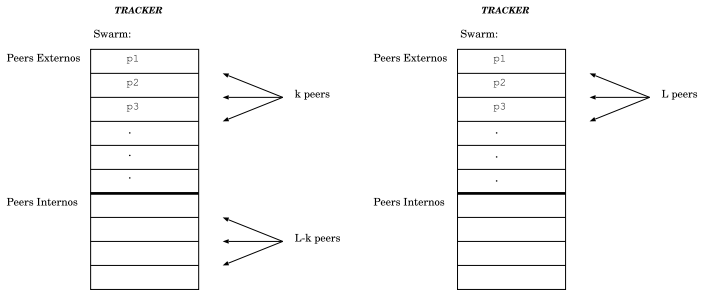
3. Proposta de um mecanismo de escolha tendenciosa

Iremos propor um mecanismo simples de escolha tendenciosa que modifica somente o *tracker* que será operado pelo ISP que deseja reduzir o tráfego BT em sua rede. Nosso mecanismo requer apenas que o *tracker* seja capaz de distinguir entre *peers* internos e externos ao ISP, o que é bem razoável, dado que o ISP está operando o *tracker* (e.g., esta distinção poderia ser feita com base na faixa de endereços IPs). Esta informação é bastante reduzida, quando comparado a propostas que requerem que os *trackers* tenham conhecimento dos diferentes ISPs dos *peers* que participam de um *swarm*. Propostas que utilizam informações mais detalhada assumem uma cooperação entre os ISPs, o que nem sempre é viável, pois ISPs em geral não querem revelar seus interesses a outros ISPs. Por fim, repare que um ISP tem todo interesse em operar um *tracker* que incorpore suas preferências, conforme sugerido em recentes trabalhos [Wang et al. 2010, Le-Blond et al. 2010], pois isto irá reduzir a quantidade de tráfego em sua rede.

Intuitivamente, o *tracker* será modificado para isolar os *peers* internos dos *peers* externos ao ISP. O objetivo é manter o tráfego P2P o máximo possível entre os *peers* internos. Entretanto, este isolamento nem sempre poderá ser total, por exemplo, quando o

seed for externo ao ISP, ou quando tivermos poucos *peers* internos ao ISP. Assumiremos que o *tracker* conhece o conjunto de *peers* internos e externos conectados no *swarm*.

No mecanismo proposto, a forma com a qual o *tracker* escolhe os vizinhos depende do *peer* ser interno ou externo. Seja L o número de vizinhos que o *tracker* deve escolher para cada *peer* que entra no *swarm*. Ao ser contactado por um *peer* externo, o *tracker* seleciona L vizinhos do conjunto de *peers* externos ao ISP, de forma aleatória com distribuição uniforme. Desta forma, um *peer* externo nunca estabelece conexão com um *peer* interno. Seja $k > 0$ um parâmetro do método que determina o número de vizinhos externos que um *peer* interno irá receber ao solicitar uma lista de vizinhos para o *tracker*. Logo, ao ser contactado por um *peer* interno, o *tracker* seleciona $L - k$ vizinhos do conjunto de *peers* internos ao ISP e outros k vizinhos do conjunto dos externos. Repare que um *peer* interno pode estabelecer conexão com um *peer* externo e eventualmente trocar blocos com ele. Por fim, a escolha dos $L - k$ vizinhos internos assim como a escolha dos k vizinhos externos continuam sendo realizada de forma aleatória com distribuição uniforme entre os *peers* de cada conjunto. Ou seja, todo *peer* interno ou externo tem igual probabilidade de ser escolhido para compor a lista de vizinhos dado seus respectivos conjuntos.



(a) *Peers* internos ao ISP. (b) *Peers* externos ao ISP.

Figura 1. Política de escolha de peers do BT modificado baseado na localização dos peers com relação ao tracker (internos ou externos ao ISP).

A figura 1 ilustra o mecanismo de escolha tendenciosa proposta. Repare que a escolha de vizinhos feita pelo *tracker* depende da localização do peer, sendo feita de acordo com a figura 1(a) caso o *peer* seja interno, ou de acordo com a figura 1(b) caso o *peer* seja externo.

4. Avaliação teórica

Faremos nesta seção uma avaliação simplificada do método de escolha de vizinhos proposto neste trabalho. Esta avaliação tem como objetivo caracterizar os vizinhos dos *peers* internos e externos, assim como os vizinhos do *seed*. Esta avaliação servirá como base para compreender os resultados obtidos com simulações detalhadas, que serão apresentados na Seção 5.

Seja n_I o número de *peers* internos ao ISP, n_E o número de *peers* externos, L o número de vizinhos da lista de *peers* retornada pelo *tracker*, e k o número de vizinhos externos que o *tracker* deve selecionar para um *peer* interno. Iremos assumir no que segue que o *swarm* possui apenas um *seed* e que este *peer* é externo ao ISP. Considere a política tendenciosa de seleção de vizinhos de um determinado *peer* e defina os seguintes eventos:

- INS: um *peer* interno não possui o *seed* como vizinho.
- ENS: um *peer* externo não possui o *seed* como vizinho.
- LNS: um *peer* não possui o *seed* como vizinho.
- NSS$_j$: j-ésimo *peer* externo ao ISP para integrar a lista de vizinhos não é o *seed*.

Estamos interessados em calcular a probabilidade de um *peer* interno ter o *seed* como vizinho, ou seja, P_I; e também a probabilidade de um *peer* externo ter o *seed* como vizinho, ou seja, P_E. Assim temos:

$$P_I = 1 - P[INS] = 1 - (P[NSS_1] \cap \ldots \cap P[NSS_k]) = \frac{k}{n_E}$$

$$P_E = 1 - P[ENS] = 1 - (P[NSS_1] \cap \ldots \cap P[NSS_L]) = \frac{L}{n_E}$$

A diferença entre P_I e P_E está no número de sorteios para *peers* externos, que é quando o *seed* pode ser selecionado, que são respectivamente, k e L. Considerando que k é menor do que L, temos que $P_I < P_E$. Esse resultado é intuitivo, já que, quanto menor o valor de k, menor será o número de vizinhos externos que um *peer* interno irá conhecer, diminuindo assim a probabilidade de um peer interno ter o *seed* como vizinho. Isto possui implicações diretas no desempenho dos *peers*, uma vez que o *seed* possui todo o conteúdo e divide sua banda igualmente entre seus vizinhos.

A partir de P_I e P_E, podemos obter o número médio de vizinhos do *seed*, uma vez que a seleção de vizinhos é independente e identicamente distribuída entre os *peers*. Desta forma, temos que o número médio de vizinhos do *seed* que são internos ao ISP é dado por $E_I = n_I\, P_I = \frac{k\, n_I}{n_E}$. De forma análoga, temos que o número médio de vizinhos do *seed* que são externos ao ISP é dado por $E_E = n_E\, P_E = L$.

Podemos ver que E_E é constante, sempre igual a L. Para os externos, quanto maior for o tamanho da lista de vizinhos, maior a chance que ela contenha o *seed*. Entretanto, E_I varia de acordo com os parâmetros do sistema. A quantidade de vizinhos internos do seed depende de k e da relação entre n_I e n_E. Quanto mais internos houver no *swarm* maior o número de internos que tem o *seed* como vizinho. Por outro lado, quanto mais externos houver, menor será o número médio de internos que tem o *seed* como vizinho. Novamente, isto possui implicação direta no desempenho dos *peers*.

Por fim, podemos somar E_E e E_I para obter o número médio de vizinhos do *seed*, ou seja $E_T = L + \frac{k\, n_I}{n_E}$. Iremos comparar este valor com o BT original, para caracterizar o número de vizinhos do *seed* em ambos os casos.

No BT original, no qual a seleção de *peers* é aleatória e uniforme, todos os *peers* têm a mesma probabilidade de ter o *seed* como vizinho. Seja P_A a probabilidade de um *peer* ter o *seed* como vizinho no BT original. De forma análoga ao desenvolvimento acima, temos que $P_A = 1 - P[LNS] = 1 - (P[NSS_1] \cap \ldots \cap P[NSS_L]) = \frac{L}{n_I + n_E}$. De forma análoga, podemos obter o número médio de vizinhos do *seed* no BT original, que é dado por $E_A = (n_I + n_E)\, P_A = L$.

Ao comparar E_T com E_A, podemos ver claramente que o *seed* possui mais vizinhos quando o *tracker* implementa a seleção de *peers* tendenciosa. Intuitivamente, isto ocorre por que peers externos só podem ter como vizinho outros peers externos, aumentando a chance do seed ser escolhido por estes.

A partir dos resultados acima, podemos calcular a fração média de banda do *seed* para todos os *peers* internos ao ISP no mecanismo tendencioso e original. Essa medida de interesse é importante, pois o *seed* além de possuir todo o conteúdo, pode ter uma velocidade de *upload* maior que dos *leechers*, influenciando dessa forma no tempo de *download*. Como o *seed* divide sua banda de *upload* de maneira uniforme entre todos os seus vizinhos, a fração média de banda que cada vizinho recebe é simplesmente dada pelo inverso do número médio de vizinhos que ele possui. No caso do BT tendencioso, temos que $B_I = E_I \frac{1}{E_T} = \frac{k\,n_I}{k\,n_I + L\,n_E}$.

No BT original, a fração de vizinhos do *seed* que são internos é dada por $n_I/(n_I + n_E)$, já que todos os *peers* são tratados igualmente. Logo, o número médio de vizinhos do *seed* que são internos, é dado por $E_A\,n_I/(n_I + n_E)$. Com isto, obtemos a fração média de banda do *seed* para cada vizinho no BT original, dada por $B_A = 1/E_A\,E_A\,n_I/(n_I + n_E)$ $= \frac{n_I}{n_I + n_E}$.

Por fim, repare que $B_I < B_A$ para alguns valores de k, n_I e n_E, em particular, quando k é pequeno ou quando $n_E > n_I$, que são casos de interesse. Nestes casos, a fração de banda que os *peers* internos recebem do *seed* será sempre menor no mecanismo tendencioso. Este fato terá impacto direto no desempenho dos *peers* internos, que será refletido no aumento do tempo médio de *download*. A próxima seção caracteriza esta e outras observações utilizando simulações detalhadas.

5. Simulador BT e medidas de interesse

Para avaliar o desempenho do BT original e a proposta para seleção tendenciosa de vizinhos, utilizamos um simulador do aplicativo desenvolvido com a ferramenta Tangram-II [Rocha et al. 2009]. O simulador desenvolvido é uma fiel implementação do protocolo de referência BT 4.0.0, contendo todos os seus mecanismos (*TFT, strict priority, rarest first, end game, optimistic unchoke, etc*), assim como troca e formato de mensagens (*choke, unchoke, have, block request, interest, network exit, peer list, etc*). Além disso, para este trabalho desenvolvemos uma nova versão do *tracker* que implementa a seleção tendenciosa de vizinhos, conforme descrito na Seção 3.

O seguinte cenário será considerado nas simulações. O *seed* é um peer externo e entra no *swarm* no instante de tempo zero e um determinado número de *leechers* internos e externos ao ISP entram no *swarm* de acordo com um processo de Poisson com média de 1/10 unidade de tempo entre chegadas. Após todas as chegadas, o *tracker* começa a atender aos pedidos de listas de vizinhos de cada *peer*. Esta decisão foi feita para considerar o cenário *flash-crowd* e permitir uma melhor avaliação do mecanismo sendo proposto. Dessa forma, todos os pedidos de lista de vizinhos que forem requisitados ao *tracker* antes da chegada de todos os *peers* serão ignorados. Por fim, os *leechers* saem do *swarm* assim que completam o *download* do último pedaço, apenas terminando de transferir os blocos pendentes em sua lista de *uploads* até o momento do término.

Consideramos o cenário sem *seeds* internos para avaliar o pior caso para o ISP. Um cenário aonde há pelo menos um *seed* interno não é muito interessante, pois esse *seed* seria capaz de servir todos os *peers* internos sozinho, sem a necessidade de adquirir informação externa. Neste caso, o ISP poderia até impor uma separação total entre os *peers* externos e internos, fazendo $k = 0$.

O simulador desenvolvido possui muitos parâmetros que podem ser instanciados para avaliar diferentes configurações do BT. Para a avaliação que segue, alguns parâmetros foram fixados, enquanto outros tiveram seus valores variados. Os parâmetros e valores utilizados estão nas Tabelas 1[1] e 2. É importante ressaltar que uma única rodada de simulação com 160 peers leva em torno de 1h em uma máquina Core i5. Desta forma, os cenários que podem ser avaliados ficam bem limitados com relação ao tamanho do *swarm*, pois em geral são necessárias muitas rodadas para garantir uma boa estimativa das medidas de interesse.

Tabela 1. Parâmetros fixos.

Parâmetros	Valores
Tamanho da lista de *peers* enviada pelo *tracker*	20
Número mínimo de vizinhos	20
Número máximo de vizinhos	80
Taxa de *upload* de *leechers*	64 Kbps
Taxa de *upload* do *seed*	200 Kbps
Tamanho do arquivo a ser disseminado	100 pedaços
Subdivisão de cada pedaço	16 blocos
Tamanho de um bloco	64 KB

Tabela 2. Parâmetros variados ao longo das simulações.

Parâmetros	Valores
k	1, 3, 5, 7, 10
Número de *leechers* no *swarm*	90, 120, 160
Proporção entre grupos (Externos – Internos)	50% + 1 seed –50%

5.1. Medidas de interesse

Para avaliar o desempenho do BT e do mecanismo de seleção tendenciosa tanto para os *peers* quanto para os ISPs, iremos considerar as seguintes medidas de interesse:

1. *Redução de tráfego*: Para avaliar a redução de tráfego inter-ISPs iremos utilizar uma métrica chamada redundância. A redundância é a média do número de vezes que um bloco entra ou sai do ISP. Assim, teremos tanto a redundância de fora para dentro do ISP como de dentro para fora. Note que todo bloco precisa necessariamente entrar ao menos uma vez no ISP, dado que o *seed* é externo ao ISP.
2. *Tempo de download*: Para avaliar o desempenho dos peers, iremos considerar o tempo médio de *download*, que é uma medida diretamente relacionada à qualidade de serviço do aplicativo. O tempo de *download* de um *peer* é dado pelo intervalo de tempo entre a primeira vez que este *peer* demonstra interesse pelo conteúdo (envia uma mensagem a um de seus vizinhos) e o momento em que ele consegue adquirir o conteúdo por completo. Para estimar o tempo médio de *download*, iremos utilizar apenas os 50% primeiros *peers* a terminarem o *download*. Isso é feito porque alguns *leechers* podem permanecer muito tempo sem possuir algum vizinho capaz de transmitir o último pedaço do conteúdo, ocasionando grande variabilidade no tempo de *download*.

[1]Como descrito em [Liogkas et al. 2006], *seeds* possuem habitualmente taxa de *upload* alta em comparação a *leechers*.

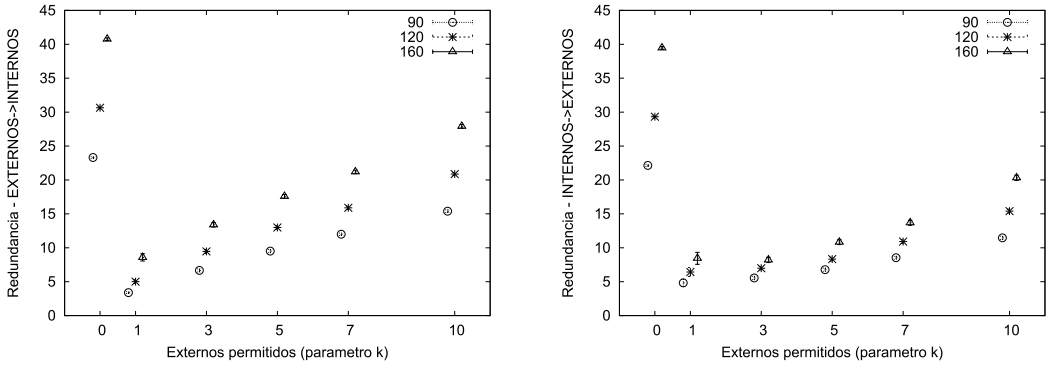
3. *Fração de término*: Para avaliar o desempenho percebido pelos usuários, iremos considerar a fração de *peers* que terminam satisfatoriamente o *download*. Nem sempre todos os *peers* irão concluir o *download*, mesmo quando temos um *seed* presente no *swarm*. Isto pode ocorrer quando ocorre uma situação de *deadlock*, que será explicada na Seção 6.3.

Para obter as medidas de interesse acima, foram realizadas 108 rodadas do simulador para cada conjunto de parâmetros. Utilizando estas rodadas, obtemos a média amostral e o intervalo de confiança de 95%, que são reportados nos gráficos a seguir.

6. Avaliação do desempenho

Nesta seção apresentamos a avaliação do mecanismo proposto e uma comparação com o mecanismo original. Um dos parâmetros fundamentais do mecanismo é o valor de k, que indica o número de vizinhos externos que cada vizinho interno recebe do *tracker* em sua lista de vizinhos. Os valores para k igual a zero se referem a resultados obtidos com o BT original (sem modificações no *tracker*). Além disso, para cada valor de k os gráficos apresentam os resultados para os três diferentes tamanhos de *swarm* considerados (90, 120 e 160 *leechers*).

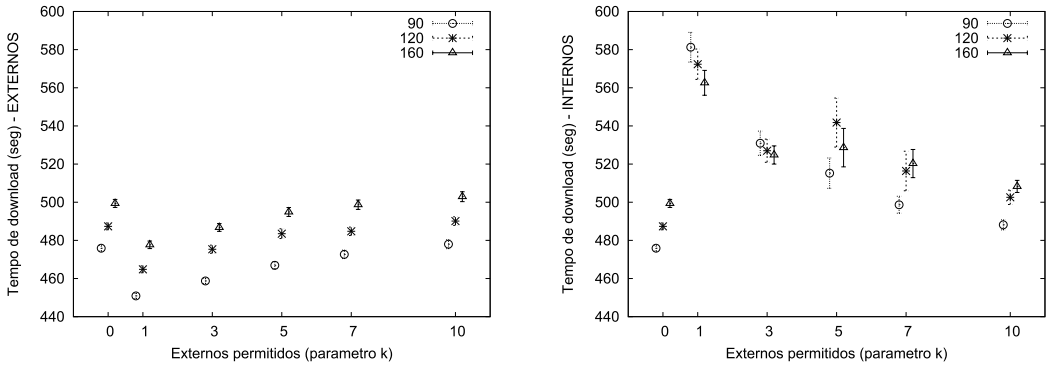
6.1. Redução de tráfego inter-ISP



(a) Direção externo → interno. (b) Direção interno → externo.

Figura 2. Redundância do tráfego no enlace inter-ISP para diferentes valores de k e diferentes tamanhos de swarm (ponto $k = 0$ representa BT original).

Os gráficos na Figura 2 apresentam os resultados da redundância do tráfego BT. Podemos notar uma grande redução na quantidade de tráfego redundante que entra e sai do ISP quando utilizamos o mecanismo de escolha tendenciosa de pares para todos os valores de k. Para valores pequenos, $k = 1$, a redução em ambos os sentidos chega a mais de 75%, trazendo grandes benefícios para o ISP. Podemos observar ainda que à medida que k aumenta, a redundância média também aumenta, o que é intuitivo, uma vez que *peers* internos terão mais vizinhos que são externos. Podemos notar ainda que a redução da redundância é maior no sentido interno → externo, ou seja mais *peers* internos deixam de enviar blocos aos *peers* externos. Isto ocorre por que o *seed* é um *peer* externo ao ISP, e necessariamente a informação precisa fluir nesta direção, além do *seed* ter uma maior capacidade de *upload* que os outros *leechers*. Em todo o caso, os resultados ilustram a vantagem que o mecanismo de seleção tendenciosa de vizinhos traz para os ISPs.



(a) Dentre os peers externos. (b) Dentre os peers internos.

Figura 3. Tempo médio de download dos peers em seus respectivos conjuntos (ponto $k = 0$ representa BT original).

6.2. Tempo de download

Os gráficos na Figura 3 apresentam o tempo médio de *download* para os *peers* externos e internos ao ISP. Considerando o conjunto de *peers* externos ao ISP (Figura 3.(a)), podemos ver que houve uma boa redução do tempo médio de *download* para valores de k pequeno independente do tamanho de *swarm* quando comparado ao BT original. Ou seja, o mecanismo acaba beneficiando os peers externos ao ISP. Entretanto, ao aumentarmos k, o tempo médio de *download* se aproxima do BT original. Intuitivamente, isto ocorre pois com o aumento de k o *seed* passa a ser mais compartilhado com *peers* internos, reduzindo a banda do *seed* para os *peers* externos, conforme vimos na análise da Seção 4.

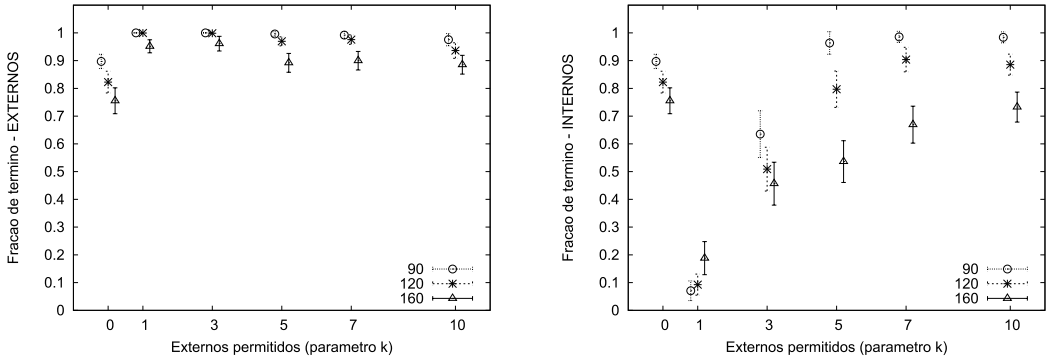
Considerando o conjunto dos *peers* internos (Figura 3.(b)), podemos observar um comportamento bem diferente. Em particular, o tempo médio de *download* aumentou significativamente para valores de k baixo, independente do tamanho do *swarm*, quando comparado com o BT original. Para valores de k mais altos ($k = 10$) este aumento é menor, mas ainda assim perceptível. Este é um resultado bastante negativo, pois o mecanismo tendencioso impõe um pior desempenho aos usuários do ISP.

Por fim, podemos observar que para k pequeno há uma inversão da relação do tempo médio de *download* e o tamanho do *swarm*. Neste caso, o tempo médio de *download* é menor para *swarms* maiores. Intuitivamente, isto ocorre pois para k pequenos é melhor que o grupo dos internos seja maior, pois isto aumenta as chances de um deles estar conectado ao *seed*, como vimos na análise da Seção 4. Note que isto não ocorre para o conjunto dos externos, onde um *swarm* maior sempre possui um maior tempo médio de *download*.

6.3. Fração de término

A fração de término do *download* poderá ser diferente de zero caso ocorra uma situação de *deadlock*. Um *deadlock* irá ocorrer quando os *leechers* formarem um componente conexo na rede de vizinhos, no qual cada um possui ao menos L vizinhos para os outros *peers*, todos têm pelo menos um pedaço em comum faltando e o *seed* não faz parte deste componente conexo. Nesse caso, cada *peer* possui o número mínimo de vizinhos estabelecido pelo protocolo e nenhum deles irão solicitar mais vizinhos ao *tracker*. Como ao

menos um pedaço está faltando na componente conexa, após certo tempo todos os *leechers* ficarão aguardando em *deadlock* pelo pedaço que nenhum deles possui, e não irão concluir o *download*. Esta situação pode ocorrer no protocolo de referência do BT e não é tratada pelo mesmo. Na prática, muitos aplicativos são modificados para poder contornar esse problema, por exemplo, solicitando mais vizinhos ao *tracker* quando a taxa de *download* vai à zero.



(a) Dentre os peers externos. (b) Dentre os peers internos.

Figura 4. Fração de peers que terminam o download com relação aos seus respectivos conjuntos (ponto $k = 0$ representa BT original).

Os gráficos ilustrados na Figura 4 apresentam as frações médias de término de *download* para os dois conjuntos de *peers* (internos e externos). Primeiramente, é importante notar que a fração de término não é igual a um nem mesmo no BT original, ocorrendo um percentual significativo de *peers* que não termina o *download* (entram em *deadlock*). Entretanto, considerando os *peers* externos (Figura 4.(a)), as frações de término obtidas mostraram-se todas aproximadamente 10% acima dos valores obtidos com o BT original, independente do valor de k. Observamos ainda que quanto maior o tamanho do *swarm* menor é a fração de término dos *peers*. Ou seja, o mecanismo de escolha tendenciosa melhorou a fração de término dos *peers* externos.

O desempenho é bem diferente quando consideramos o conjunto dos internos (Figura 4.(b)). Neste caso, a fração de término é bem menor, chegando a apenas 20% para $k = 1$. Para maiores *swarms* (160 peers), a fração de término dos internos é sempre pior do que o BT original. Observamos ainda que quanto maior o *swarm* menor é a fração de término, assim como no caso dos externos, mas de forma mais acentuada. Este certamente é um resultado indesejado para o para o mecanismo de escolha tendenciosa. Na próxima seção veremos como este efeito pode ser mitigado, inclusive para o BT original.

7. Modificando o BitTorrent

Vimos na seção anterior que até mesmo o BT original possui um baixo desempenho com relação à fração de *peers* que terminam o *download* por completo, problema que é exacerbado para *peers* internos quando utilizamos uma política tendenciosa para seleção de *peers*. Parte do problema está em os *peers* saírem do *swarm* assim que terminam de baixar o último pedaço, não dando chances deste pedaço ser servido pelo *peer*. Tendo em vista esta causa, propomos uma simples modificação no protocolo que irá obrigar os *peers* a permanecerem no *swarm* e servirem outros *peers* por um pequeno período de tempo antes

de deixarem o *swarm*. Nossa proposta consiste de dois simples critérios para determinar este período:

- *Timeout*: Se após o tempo necessário para transmitir um pedaço completo, nenhum "novo" pedaço for solicitado ao *peer*, o mesmo pode sair do *swarm*. Um "novo" pedaço é um pedaço que não tenha sido servido por esse *peer* enquanto ele era um *leecher* (ou seja, antes dele completar o *download* do último pedaço). Repare que este tempo é o tempo de transmissão de um pedaço.
- *Após servir um novo pedaço*: Caso algum novo pedaço tenha sido requisitado ao *peer* antes do *timeout*, o mesmo poderá sair da rede assim que receber a confirmação que este mesmo pedaço está disponível em algum outro vizinho. Neste momento, o *peer* pode sair do *swarm*. No pior caso, o *peer* transmite o pedaço por inteiro.

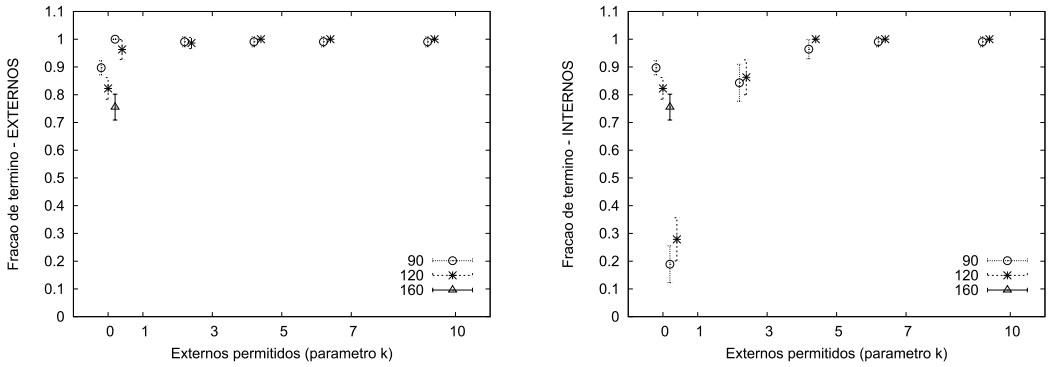
A viabilidade técnica de implementar esta proposta está baseada em um sistema de incentivo que o ISP pode oferecer aos seus clientes caso utilizem esta versão do protocolo, como por exemplo, incentivo econômico (redução de tarifas mensais) e incentivo de banda, aumentando a capacidade de *download* e *upload* do *peer*. Dessa forma, além de contribuir para o desempenho global do *swarm*, os *peers* poderão obter vantagens individuais, permanecendo conectados um tempo adicional muito pequeno em relação ao tempo total de *download*.

Intuitivamente, o mecanismo acima dá oportunidade aos vizinhos do *peer* de solicitar e obter o último pedaço antes do mesmo sair do *swarm*, o que pode ter efeitos drásticos. Ao adquirir um bloco que nenhum outro *peer* em sua vizinhança possui e permanecer no *swarm* por mais tempo, seus vizinhos terão tempo de demonstrar interesse neste bloco. Os que estiverem na condição *unchoke*, e anteriormente não estavam recebendo blocos de outro pedaço, irão requisitar o último pedaço, por ser o mais raro. Dessa forma, o *peer* irá repassar o pedaço a outros *peers* antes da sua saída, garantindo assim ao menos mais uma réplica deste raro pedaço em sua vizinhança. Repare que estes outros *peers* que adquirirem este pedaço irão contribuir para sua disseminação, passando pelo mesmo procedimento, se este também for seu último pedaço.

Ao permitir que um *peer* saia do *swarm* assim que este adquirir um último pedaço, seus vizinhos serão obrigados a aguardar que outro *peer* da vizinhança obtenha este pedaço. Se este outro *peer* procede da mesma maneira, a situação se repete e o problema pode se agravar, dando origem ao *deadlock* descrito na Seção 6.3. Desta forma, um fenômeno relacionado com este problema, também causado pela sincronização entre os pedaços faltantes nos *peers*, foi identificado e avaliado teoricamente em um recente artigo [Hajek and Zhu 2010].

7.1. Avaliação da proposta

Os gráficos da Figura 5 ilustram a fração de término de *download* dos *peers* quando os mesmos implementam a modificação proposta para o protocolo. A avaliação a seguir compara os resultados anteriores (apenas mudança do *tracker*) com a proposta acima, na qual *peers* permanecem no *swarm* por um pequeno período de tempo após terminarem o *download*. Podemos notar que em todos os casos, a fração de término de *download* é maior, sendo a diferença notável (até 50% melhor) no caso do conjunto dos internos. Repare que para $k = 5$, a fração de término de *peers* internos e externos já é maior do que no próprio BT original (comparar com a Figura 4).



(a) Dentre os peers externos. (b) Dentre os peers internos.

Figura 5. Fração de peers que terminam o download com o BT modificado, no qual peers permanecem no swarm um pequeno tempo após completarem o download (ponto $k = 0$ representa BT original).

O tempo médio de *download* dos *peers* internos e externos quando a proposta acima é considerada se mostrou praticamente idêntico aos obtidos sem a proposta. Entretanto, este fato era esperado uma vez que a proposta acima obriga aos *peers* a permanecerem no *swarm* por apenas um pequeno período de tempo a mais, bem menos de 1% do tempo médio de *download*. A redundância média também não foi afetada pela proposta acima, uma vez que apenas mais uns poucos blocos serão trocados pelos *peers*. Por falta de espaço, não apresentamos estes resultados.

8. Conclusão

Neste trabalho propomos um mecanismo simples para escolha tendenciosa de peers que pode ser implementado no *tracker* e operado pelo ISP que deseja reduzir o tráfego P2P em seus enlaces inter-ISPs. Nossa proposta é uma variação mais realista da proposta de Bindal et. al, pois esta última requer a cooperação entre os ISPs para ser implementada [Bindal et al. 2006]. O mecanismo proposto utiliza apenas a distinção entre *peers* internos e externos ao ISP e não requer nenhuma outra informação. Intuitivamente, o mecanismo proposto isola parcialmente os *peers* internos dos *peers* externos.

Fazemos uma avaliação teórica simplificada do mecanismo assim como uma avaliação usando um simulador detalhado e fiel ao protocolo BitTorrent (todos os mecanismos e mensagens do protocolo BitTorrent de referência foram implementados). Resultados da simulação mostram que o mecanismo proposto reduz significativamente o tráfego inter-ISP gerado pelo BitTorrent, chegando a 75%, em alguns casos. Por outro lado, os *peers* internos ao ISP podem ter uma perda significativa de desempenho, tanto com relação ao tempo de *download* quanto à fração de término de *download*. Em ambos os casos os *peers* internos são prejudicados com relação aos *peers* externos ao ISP, resultado indesejado para o ISP e seus clientes. Para mitigar este problema, propomos uma simples modificação do protocolo BitTorrent de forma que *peers* permaneçam no *swarm* um pequeno intervalo de tempo após completarem o *download*. Resultados desta proposta mostram que o desempenho dos *peers* melhora significativamente, principalmente a fração de término de *download*, exigindo apenas que os *peers* permaneçam o tempo de transmissão de um pedaço, o que é mais 10 vezes menor que o tempo de médio de *download*.

Por fim, este trabalho ilustra como políticas para seleção de vizinhos podem ser vantajosas tanto para os ISPs, com relação à gerência do tráfego na rede, quanto para os seus clientes, que podem continuar a terem um bom desempenho de seus aplicativos. Esta é uma questão central a atual discussão sobre aplicativos P2P e ISPs.

Referências

Aggarwal, V., Feldmann, A., and Scheideler, C. (2007). Can ISPs and P2P users cooperate for improved performance? *ACM SIGCOMM Computer Communication Review.*

Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., and Zhang, A. (2006). Improving traffic locality in BitTorrent via biased neighbor selection. In *IEEE International Conference on Distributed Computing Systems*.

Choffnes, D. R. and Bustamante, F. E. (2008). Taming the torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *ACM SIGCOMM*.

H. Schulze and K. Mochalski (2009). Internet Study 2008/2009. Technical report, Ipoque. URL http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

Hajek, B. and Zhu, J. (2010). The missing piece syndrome in peer-to-peer communication. *CoRR*, abs/1002.3493.

Le-Blond, S., Legout, A., and Dabbous, W. (2010). Pushing bittorrent locality to the limit. *CoRR*, abs/1011.1892.

Lin, M., Lui, J. C. S., and Chiu, D.-M. (2010). An isp-friendly file distribution protocol: Analysis, design, and implementation. *IEEE Trans. on Par. and Dist. Sys.*

Liogkas, N., Nelson, R., Kohler, E., and Zhang, L. (2006). Exploiting BitTorrent for Fun (But Not Profit) *IPTPS - International workshop on Peer-To-Peer Systems*

Rocha, A., Jaime, G., Murai, F., Alves, B., Figueiredo, D., Leão, R. M. M., and de Souza e Silva, E. A. (2009). Novas evoluções integradas à ferramenta Tangram-II v3.1. In *Salão de Ferramentas / XXVII Simpósio Brasileiro de Redes de Computadores*.

Saleh, O. and Hefeeda, M. (2006). Modeling and caching of peer-to-peer traffic. In *IEEE International Conference on Network Protocols*.

Sandvine Co. (2004). Meeting the challenge of today's evasive P2P traffic. Technical report, Waterloo, Canada. An Industry Whitepaper.

Shen, G., Wang, Y., Xiong, Y., Zhao, B. Y., and Zhang, Z.-L. (2007). HPTP: Relieving the tension between ISPs and P2P. In *Int. Workshop on Peer-To-Peer Systems (IPTPS)*.

Wang, H., Liu, J., Chen, B., Xu, K., and Ma, Z. (2010). On tracker selection for peer-to-peer traffic locality. In *Peer-to-Peer Computing'10*

Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y., and Silberschatz, A. (2008). P4P: provider portal for applications. In *ACM SIGCOMM*.